ISRN SICS/R--91/05--SE

On the Complexity of Equation Solving in Process Algebra

by

Bengt Jonsson and Kim Guldstrand Larsen

SICS research report
R91:05
ISSN 0283-3638

On the Complexity of Equation Solving in Process Algebra *

Bengt Jonsson †
Swedish Institute of Computer Science ‡
and Dept. of Computer Systems, Uppsala University

Kim Guldstrand Larsen
Aalborg University, Denmark §

SICS Research Report 91:05
February 1991

Abstract

The problem of designing a system which in a given environment C should satisfy a given specification S can be formulated as "find a system P such that $C(P)$ satisfies the specification S". In process algebra, such problems take the form of equations. We investigate the complexity of solving such equations in process algebra. We consider the problem of deciding whether there is a process P which satisfies an equation of one of the following forms:

$$C(P) \sim Q \qquad C(P) \lhd S \qquad (A \mid P)\backslash L \sim B \qquad (A \mid P)\backslash L \approx B$$

where C is an arbitrary context of some process algebra, A, B and Q are given processes, S is a modal specification, $\sim$ ($\approx$) is (weak) bisimulation equivalence, $\lhd$ is refinement between modal specifications (a generalization of bisimulation equivalence), and $\mid$ and $\backslash L$ is the parallel and restriction operator of CCS respectively. The main result is that all four problems are PSPACE-hard in the size of the given contexts, processes and specifications. The four problems are still PSPACE-hard if the right-hand side of the equation is required to be deterministic and the number of involved actions is bounded by a small constant. We also give constraints under which the first and third problem can be solved in polynomial time.

*This research report is a revised and extended version of a paper that will appear under the same title in the *Proceedings of the Colloquium on Trees and Algebra in Programming*, Brighton, England, April, 1991, published in Lecture Notes in Computer Science by Springer Verlag

†supported in part by the Swedish Board for Technical Development (STU) under contract No. 89-01220P as part of Esprit BRA project SPEC, No. 3096

‡Full Address: Bengt Jonsson, Swedish Institute of Computer Science, Box 1263, 164 28 Kista, Sweden. E–mail: bengt@sics.se

§Full Address: Kim Guldstrand Larsen, Aalborg University, Department of Mathematics and Computer Science, Fredrik Bajersvej 7, 9220 Aalborg, Denmark. E–mail: kgl@iesd.auc.dk

Introduction

One of the most difficult and important problems in computer science is to develop methods for design and construction of concurrent systems. One approach towards automatic design methods is to formulate the design problem as a model construction problem, "find a system P which satisfies a specification S," for which automatic decision procedures can be found. The specification S can for instance be a formula in temporal logic as in [MW84, PR89, CE82].

In this paper, we consider the case where the specification S does not specify the system P directly, but rather P placed in a given environment. Process algebra provides an elegant way to represent such environments formally as contexts. The design problem is then formulated as the problem of finding a system P which satisfies

$$C(P) \textbf{ sat } S \tag{1}$$

where C is a context representing the given environment, and **sat** is a suitably chosen satisfaction relation. As an example, S can be an abstract system and $C(P)$ can be a system in which P is an unknown component executing in parallel with several other known components, represented by the system A. This problem can be formulated in CCS [Mil89] as finding P which satisfies

$$(A \mid P)\backslash L \textbf{ sat } S \tag{2}$$

Here L is the set of actions over which A and P communicate. The operation $\mid$ puts two systems in parallel, and the operation $\backslash L$ makes the actions in the set L internal and unobservable.

Methods for solving (2) have been presented by Shields [Shi89], by Parrow [Par89a], by Lewis and Qin [QL90] and by Merlin and Bochmann [MB83]. For the more general problem (1), Larsen and Xinxin [LX90b] have developed a language-independent theory of contexts in process algebra, in which they give a characterization of the solutions of (1) with **sat** being bisimulation equivalence, which induces a single exponential time decision algorithm.

Common to all proposed methods [Shi89, Par89a, MB83, LX90b, QL90] is that the proposed algorithms require exponential time, even though restrictions are imposed on the involved contexts and processes. No restrictions have been presented under which the problem can be solved in polynomial time, and there have not been presented any lower bounds on the complexity of the problem.

In this paper, we establish lower bounds on the complexity of solving equations of form (1) and (2), and also present some restrictions under which the equations can be solved in polynomial time. We consider three different satisfaction relations **sat** in (1): bisimulation equivalence, $\sim$, weak bisimulation or observational equivalence, $\approx$, and modal refinement, $\lhd$. In the two first relations, the specification S is itself a system, and in the last relation, S is a modal specification. The equivalences $\sim$ and $\approx$ are well-established and often used satisfaction relations for the correctness of concurrent systems [BK86, Koo85, LM87, Par89b, SFD85]. Modal refinement $\lhd$ is a generalization of bisimulation equivalence: a modal specification can distinguish between mandatory and optional transitions, thus allowing more loose specifications [LT88, HL89, Lar89, BL90].

The main decision problems that we consider are the following:

CCSEQ Given (finite–state) systems A and B and a set of actions L, does there exist any process P satisfying the equation $(A \mid P)\backslash L \sim B$.

CCSOBS is the same as CCSEQ but for observation equivalence $\approx$.

EQ Given a (finite–state) context C and a (finite–state) process Q, does there exist any process P satisfying the equation $C(P) \sim Q$?

INEQ Given a (finite–state) context C and a (finite–state) modal specification S, does there exist any process P satisfying the inequation $C(P) \lhd S$?

Also, we are concerned with the identification of subclasses of the above problems that are solvable in polynomial time. The results concerning these problems that we obtain are:

- The problems CCSEQ and CCSOBS are PSPACE-hard in the size of A and B. even in the case where B is deterministic and the number of involved actions is bounded by a small constant.
- The problems EQ and INEQ are PSPACE-hard in the size of C and S (C and Q for EQ), even in the case where S (Q in EQ) is deterministic and the number of involved actions is bounded by a small constant.
- Under certain conditions on the context C, we obtain a subproblem of EQ which is solvable in polynomial time. This also yields conditions under which the problem CCSEQ is solvable in polynomial time.

The lower bounds are obtained through a series of reductions from the known PSPACE–complete graph theoretical problem called Generalized Geography, GENGEO , in [GJ79]. The series of reductions is the following:

$$\text{GENGEO} \longrightarrow \text{INEQ} \longrightarrow \text{EQ} \qquad \text{INEQ} \longrightarrow \text{CCSEQ} \longrightarrow \text{CCSOBS}$$

In the actual reductions we will use a subproblem of INEQ obtained by imposing restrictions on the involved contexts and modal specifications. A deterministic exponential time upper bound for all problems can be obtained from the solution method in [LX90b]. It still remains an open problem whether these problems are in PSPACE or not.

The remainder of the paper is organized as follows. In the next section, we introduce the framework of processes, contexts and bisimulations. Section 2 introduces modal specifications and refinement. Section 3 presents the INEQ problem and states that it is PSPACE–hard. The actual proof of this fact is found in the appendix, since the GENGEO problem and its reduction to INEQ are not needed for understanding the remainder of the paper. Section 4 contains the reduction from INEQ to EQ . Section 5 contains the reduction from EQ to CCSEQ . In Section 6 we prove that the problems are PSPACE-hard also when the number of allowed actions are bounded by a small constant. Section 7 presents a polynomial time solution to a subproblem of EQ and constraints under which CCSEQ can be solved in polynomial time. Finally we discuss open problems and future work. The appendix contains the proof that INEQ is PSPACE–hard.

1 Processes, Bisimulation and Contexts

In this paper we follow the *reactive* view of concurrent systems advocated by Pnueli [Pnu85]. We describe concurrent systems (or *processes*) in terms of their interaction with their environment using the well–established model of *labelled transition systems* [Plo81].

Definition 1.1 *A* labelled transition system *is a structure* $\mathcal{P} = (W, A, \longrightarrow)$ *where* W *is a set of processes (states or configurations),* A *is a set of actions, and* $\longrightarrow \subseteq W \times A \times W$ *is a transition relation.*

Notation 1.2 Let $\mathcal{P} = (W, A, \longrightarrow)$ be a labelled transition system. A *derivation sequence* d is a finite or infinite sequence of transitions of the form:

$$d = P_0 \xrightarrow{a_0} P_1, P_1 \xrightarrow{a_1} P_2, P_2 \xrightarrow{a_2} P_3, \ldots$$

which we shall often abbreviate to:

$$d = P_0 \xrightarrow{a_0} P_1 \xrightarrow{a_1} P_2 \xrightarrow{a_2} P_3 \xrightarrow{a_3} \cdots$$

We say that P_0 is the initial process of the sequence d or that d is a derivation sequence of P_0. Whenever a process Q appears in some derivation sequence of P we say that Q is a *derivative* of P. We say that P is *finite–state* in case the set of its derivatives is finite. For $L \subseteq A$, we say that Q is *L–reachable from P*, if there exists a (finite) derivation sequence with P as initial process, Q as final process and with all actions occurring in the sequence belonging to the set L.

For $d = P_0 \xrightarrow{a_0} P_1 \xrightarrow{a_1} P_2 \xrightarrow{a_2} P_3 \xrightarrow{a_3} \cdots$ a derivation sequence we denote by $\mathrm{Proc}(d)$ the sequence of processes $P_0P_1P_2P_3\ldots$ and by $\mathrm{Act}(d)$ the sequence of actions $a_0a_1a_2a_3\ldots$.

A *computation* of a process P is a derivation sequence with P as initial process and which is maximal under the prefix ordering. Hence, the last process of any *finite* computation must be dead–locked with respect to any action. We shall use the notation $\mathrm{Comp}(P)$ to denote the set of computations of P. □

The notion of *bisimulation* [Par81, Mil83] provides a means of identifying processes based on their operational behaviour. In particular, processes at different descriptive levels of abstraction may be compared.

Definition 1.3 *Let $\mathcal{P} = (W, A, \longrightarrow)$ be a labelled transition system. Then a bisimulation $\mathcal{B}$ is a binary relation on W such that whenever $(P, Q) \in \mathcal{B}$ and $a \in A$ then the following holds:*

1. *Whenever $P \xrightarrow{a} P'$, then $Q \xrightarrow{a} Q'$ for some Q' with $(P', Q') \in \mathcal{B}$,*
2. *Whenever $Q \xrightarrow{a} Q'$, then $P \xrightarrow{a} P'$ for some P' with $(P', Q') \in \mathcal{B}$*

P and Q are said to be bisimilar in case (P, Q) is contained in some bisimulation $\mathcal{B}$. We write $P \sim Q$ in this case.

A straightforward generalization allows us to compare processes from different transition systems (essentially by applying the above definition to disjoint sums of transition systems). Bisimulation treats all actions in A equally. One sometimes wants to distinguish between observable and unobservable actions, and define an analogous equivalence in which only the observable actions of transitions are significant. This is achieved by assuming that the action set A contains an *unobservable action* τ. A labelled transition system $\mathcal{P} = (W, A, \longrightarrow)$ now induces an *observational* transition system $\mathcal{P}_O = (W, (A\backslash\{\tau\}) \cup \{\epsilon\}, \Longrightarrow)$, where $P \xRightarrow{\epsilon} Q$ if and only if there is a (possibly empty) sequence $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q$ and $P \xRightarrow{a} Q$ if and only if $P \xRightarrow{\epsilon} P_1 \xrightarrow{a} P_2 \xRightarrow{\epsilon} Q$ for $a \in A\backslash\{\tau\}$. We say that $\mathcal{B}$ is a *weak bisimulation* in case $\mathcal{B}$ is a bisimulation with respect to $\mathcal{P}_O$. We write $P \approx Q$ whenever (P, Q) is contained in some weak bisimulation. The equivalence $\approx$ is often referred to as *observational equivalence*. A process P is said to be *stable* if it cannot perform the unobservable action τ. A process P is said to be *rigid* if all its derivatives are stable.

Process algebra [Mil80, Mil89, Hoa85, BK85, Bou85] provides a framework for describing both the modular structure and the operational behaviour of reactive systems (or processes). In particular, a process algebra enables processes to be constructed (syntactically) through a number

of operators (normally including some operator for parallel composition). Semantically, these operators are described through a number of inference–rules from which the operational behaviour of a composite process may be inferred from that of its components. In Figure 1 is shown the well–known inference rules for the parallel composition operator $|$ and the restriction operator $\backslash L$ ($L \subseteq A$) of CCS.

$$\frac{P \xrightarrow{a} P'}{P \,|\, Q \xrightarrow{a} P' \,|\, Q} \qquad \frac{Q \xrightarrow{a} Q'}{P \,|\, Q \xrightarrow{a} P \,|\, Q'} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'} \qquad \frac{P \xrightarrow{a} P'}{P\backslash L \xrightarrow{a} P'\backslash L}\, a, \overline{a} \notin L$$

Figure 1: Inference rules for $|$ and $\backslash L$ of CCS

In process algebra, derived operators (or *contexts*) are normally represented syntactically as terms with free variables possibly occurring. In order to facilitate a general investigation of the problem of equation solving, we introduced in [LX90a, LX90b] an operational theory of contexts in terms of action transducers. That is, a (unary) context is semantically viewed as an object which consumes actions provided by its internal process and in return produces actions for an external observer. We shall allow transductions in which the context produces actions on its own without involving the inner process, and also, we shall assume that the context may change during transductions.

Definition 1.4 *A* context system $\mathcal{C}$ *is a structure* $\mathcal{C} = (K, A, \longrightarrow)$, *where* K *is a set of* contexts, A *is a set of* actions, $\longrightarrow \subseteq K \times (A_0 \times A) \times K$ *is a* transduction *relation,* $A_0 = A \cup \{0\}$ *with* 0 *being a distinguished* no–action *symbol (i.e.* $0 \notin A$*).*

For $(C, (a, b), C') \in \longrightarrow$ we shall adopt the notation $C \xrightarrow[a]{b} C'$ and interpret this as: "by consuming the action a the context C can produce the action b and change into C'". For $a = 0$ the production of b does not involve consumption of any action.

Example 1.5 Consider the CCS context $P \,|\, [\,]$ (we use $[\,]$ as a free variable as this notation suggests the existence of a hole in which to place an argument process). The first inference rule of Figure 1 indicates that $P \,|\, [\,]$ may produce an action without consulting its argument process Q whenever P has transitions. Stated in terms of transductions, this can be expressed as $P \,|\, [\,] \xrightarrow[0]{a} P' \,|\, [\,]$ whenever $P \xrightarrow{a} P'$. The second inference rule of Figure 1 allows the inner process to interact directly with the environment without involving the context. As a transduction we have $P \,|\, [\,] \xrightarrow[a]{a} P \,|\, [\,]$ for any action a. Finally, the third inference rule of Figure 1 indicates that the context may produce a τ–action as a result of an internal communication between the inner process (contributing $\overline{a}$) and the process P (contributing a). As a transduction this becomes $P \,|\, [\,] \xrightarrow[\overline{a}]{\tau} P' \,|\, [\,]$ whenever $P \xrightarrow{a} P'$.

Now consider a restriction context $[\,]\backslash L$. Then the inference rule given for restriction in Figure 1 may be represented as the transductions $[\,]\backslash L \xrightarrow[a]{a} [\,]\backslash L$ whenever $a, \overline{a} \notin L$. □

Now, given a (unary) context C and a process P we may syntactically form the *combined process* $C(P)$ by substituting P for the free variable (normally denoted $[\,]$) in C. The semantics of $C(P)$ is such that if $P \xrightarrow{a} P'$ and C has an a–consuming transduction $C \xrightarrow[a]{b} C'$ then $C(P) \xrightarrow{b} C'(P')$

should hold. Also, whenever $C \xrightarrow[0]{b} C'$, i.e., C has a transduction which does not involve any consumption, we require the transition $C(P) \xrightarrow{b} C'(P)$. Extending the transition relation for processes by letting $P \xrightarrow{0} Q$ if and only if $P = Q$ [1], the above expectations are both met by the following (single) inference rule for combined processes:

$$\frac{C \xrightarrow[a]{b} C' \qquad P \xrightarrow{a} P'}{C(P) \xrightarrow{b} C'(P')} \tag{3}$$

For a combined process of the form $D(C(P))$, a combined context $D \circ C$ may be defined from D and C (see [LX90a]) such that $D(C(P)) = (D \circ C)(P)$.

The *sort* of a process P, denoted sort(P) is the set of actions a such that $Q \xrightarrow{a} Q'$ for some derivatives Q and Q' of P. If there is a sequence of transductions

$$C \xrightarrow[a_1]{b_1} C_1 \xrightarrow[a_2]{b_2} C_2 \xrightarrow[a_3]{b_3} \cdots \xrightarrow[a]{b} C'$$

then we say that a is in the *inner sort* of C and that b is in the *outer sort* of C. The *sort* of C is the union of the inner and outer sort of C.

2 Modal Transition Systems and Refinement

Modal Transition Systems provides a (graphical) specification formalism for processes and is studied at length in [LT88, HL89, Lar89, BL90, LX90b]. By a graphical specification formalism we mean a formalism based on transition systems, in contrast to logical formalisms. The specifications expressible using Modal Transition Systems (Modal Specifications) typically impose restrictions on the transitions of possible implementations by telling which transitions are *necessary* and which are *admissible*. This is reflected by the structure of a Modal Transition System which contains *two* transition relations: $\longrightarrow_\Box$ for describing the *required* transitions and $\longrightarrow_\Diamond$ for describing the *allowed* transitions.

Definition 2.1 *A* modal transition system *is a structure* $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, *where* Q *is a set of modal specifications,* A *is a set of actions and* $\longrightarrow_\Box, \longrightarrow_\Diamond \subseteq Q \times A \times Q$ *are two transition relations satisfying the condition* $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$.

The condition $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ says that anything required is also allowed, ensuring that any modal specification is consistent. A modal specification S is *deterministic* if $T \xrightarrow{a}_\Diamond T_1$ and $T \xrightarrow{a}_\Diamond T_2$ implies $T_1 = T_2$ whenever T is a derivative of S. The sort of a modal specification is defined analogously as the sort a process. For a standard labelled transition system $\mathcal{P} = (W, A, \longrightarrow)$, we may consider the derived modal transition system $\mathcal{S} = (W, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, with $\longrightarrow_\Box = \longrightarrow_\Diamond = \longrightarrow$; i.e. we view processes as modal specifications where all transitions are both allowed and required.

Now, the more a specification requires and the less it allows the stronger we expect the specification to be. Using the transition relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ this may be formalized by the following notion of *refinement*.

Definition 2.2 *Let* $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ *be a modal transition system. A* refinement $\mathcal{R}$ *is a binary relation on* Q *such that whenever* $(S, T) \in \mathcal{R}$ *and* $a \in A$ *then the following holds:*

[1] Note, that this extension does *not* change which processes are bisimular.

1. *Whenever* $S \xrightarrow{a}_{\Diamond} S'$, *then* $T \xrightarrow{a}_{\Diamond} T'$ *for some* T' *with* $(S', T') \in \mathcal{R}$,

2. *Whenever* $T \xrightarrow{a}_{\Box} T'$, *then* $S \xrightarrow{a}_{\Box} S'$ *for some* S' *with* $(S', T') \in \mathcal{R}$.

S is said to be a refinement *of T in case (S, T) is contained in some refinement R. We write $S \lhd T$ in this case.*

As for bisimulation the notion of refinement may be generalized so that specifications from different modal transition systems can be compared. If $P \lhd S$, where P is a process (viewed as a specification through the derived modal transition system) and S is a specification, we will say that P is an *implementation* of S. For P and Q processes it is easy to see that $P \lhd Q$ becomes equivalent to $P \sim Q$.

3 Inequation Solving

The problem of Inequation Solving INEQ is defined as follows:

Instance: A finite collection of pairs

$$\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$$

where I is a finite index set, and for all $i \in I$, C_i is a finite–state context and S_i is a finite–state modal specification.

Question: Does there exist a process P such that the inequation $C_i(P) \lhd S_i$ is satisfied for all $i \in I$?

The size of an instance is the sum of the number of states, transitions and transductions in the contexts and specifications $\{(C_i, S_i) \mid i \in I\}$. Let INEQ^1 be the subproblem of INEQ where only singleton collections are allowed. Despite the restriction, it turns out that any instance $\{(C_i, S_i) \mid i \in I\}$ of INEQ can be transformed (in polynomial time) into an equivalent instance (C, S) of INEQ^1 : simply let $C \xrightarrow[0]{a_i} C_i$ and $S \xrightarrow{a_i}_{\Box} S_i$ for all $i \in I$ (assuming that the actions a_i are all different), then it is easy to see that the solutionsets coincide:

$$\{P \mid \forall i \in I \,.\, C_i(P) \lhd S_i\} = \{P \mid C(P) \lhd S\}$$

Let INEQ$_d$ be the subproblem of INEQ , where the modal specifications of an instance are all required to be deterministic.

Theorem 3.1 *The decision problems* INEQ *and* INEQ$_d$ *are both PSPACE–hard.*

Proof: As announced in the Introduction, we have transferred the proof of this theorem to the Appendix. The proof is a reduction from the known PSPACE–complete graph theoretical problem called Generalized Geography, GENGEO , in [GJ79]. □

From the remark in the beginning of this section it follows that also the decision problem INEQ^1 is PSPACE–hard.

In the following sections, we shall prove PSPACE-hardness for other decision problems by reduction from the problem INEQ . In some cases, we need first to impose restrictions on the problem INEQ before the reduction. In the remainder of this section, we shall define these restrictions.

Definition 3.2 *Let $\mathcal{C} = (K, A, \longrightarrow)$ be a context system and let $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ be a modal transition system. A* correspondence relation *$\propto$ is a binary relation between K and Q such that whenever $(C, S) \in \propto$ and $a \in A \cup \{0\}$ and $b \in A$ then*

1. $C \xrightarrow[a]{b} C'$ *and* $S \xrightarrow{b}_\Diamond S'$ *imply* $(C', S') \in \propto$,
2. $(C, S) \in \propto$ *and* $(C, S') \in \propto$ *imply* $S = S'$.

A relation $\propto$ between K and Q is safe *if whenever $(C, S) \in \propto$ and $S \xrightarrow{b}_\Box S'$ then either*

1. *there is exactly one transduction from C, or*
2. *for each $a \in A$ there is a C' with $C \xrightarrow[a]{b} C'$, and $S \xrightarrow{b}_\Box S'$ is the only transition from S (except the implied $S \xrightarrow{b}_\Diamond S'$).*

Let INEQ_R be the subproblem of INEQ_d where in each instance $\{(C_i, S_i) \mid i \in I\}$ we require that (C_i, S_i) be contained in a safe correspondence relation for each $i \in I$. Let INEQ_R^1 be the subproblem of INEQ_R where only singleton collections are allowed.

Theorem 3.3 *The decision problems INEQ_R and INEQ_R^1 are PSPACE–hard.*

Proof: Follows from the proof of Theorem 3.1, by inspecting the constructed instances of INEQ , and checking that they are also in INEQ_R . PSPACE-hardness of INEQ_R^1 follows from the remark in the beginning of this section, in the same way as for INEQ^1 . □

We shall also need the following small lemma.

Lemma 3.4 *Let $\mathcal{C} = (K, A, \longrightarrow)$ be a context system and let $\mathcal{S} = (Q, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$ be a modal transition system. If $\propto$ is a correspondence relation, then $\{(C(P), S) \mid C(P) \lhd S$ and $(C, S) \in \propto\}$ is a refinement.*

Proof: trivial. □

4 Equation Solving

The problem of equation solving EQ is the subproblem of INEQ obtained by restricting the instances to be collections $\{(C_i, Q_i) \mid i \in I\}$ where Q_i is a *process* for all $i \in I$ (recall that any process can be viewed as a modal specification). The problem of INEQ then reduces to whether there exists a process P satisfying the *equation* $C_i(P) \sim Q_i$ for all $i \in I$.

The following lemma provides the basis for establishing PSPACE–hardness of EQ :

Lemma 4.1 *Let S be a deterministic, finite–state modal specification. Then there exists a context C_S and a process Q_S such that for all processes P with $sort(P) \subseteq sort(S)$ the following equivalence holds:*

$$P \lhd S \;\Leftrightarrow\; C_S(P) \sim Q_S$$

Moreover, the size of both C_S and Q_S is quadratic in the size of S, and Q_S is deterministic.

Proof: For each specification S we define contexts C_S, D_S and a process Q_S. We state the inference rules defining the behaviours of C_S, D_S, and Q_S in terms of that of S:

$$\frac{S \xrightarrow{a}_{\Diamond} S'}{C_S \xrightarrow[a]{a} C_{S'}} \qquad \frac{S \xrightarrow{a}_{\Diamond} S' \quad S \not\xrightarrow{a}_{\Box} S'}{C_S \xrightarrow[0]{a} D_{S'}} \qquad \frac{S \not\xrightarrow{a}_{\Diamond}}{C_S \xrightarrow[a]{x}} \qquad \frac{S \xrightarrow{a}_{\Diamond} S'}{D_S \xrightarrow[0]{a} D_{S'}} \qquad \frac{S \xrightarrow{a}_{\Diamond} S'}{Q_S \xrightarrow{a} Q_{S'}}$$

where a ranges over the actions in sort(S) and x is a new action symbol. The idea is that C_S is a context which behaves like its inner process P (by the leftmost rule). However, in case S allows a transition but does not require it, then C_S must be able to perform that transition even when its inner process cannot. This is attained by the transition to some $D_{S'}$ (the second rule), whereafter $D_{S'}$ behaves exactly like S'. Finally, C_S prohibits disallowed moves of P by translating them to some distinguished action x which is not in the sort of S.

For the $\Rightarrow$–direction, we show that $\mathcal{B} = \{(C_S(P), Q_S) \,|\, P \lhd S\} \cup \{(D_S(P), Q_S)\}$ is a bisimulation. That the pairs of form $(D_S(P), Q_S)$ are contained in a bisimulation is obvious from the fourth and fifth inference rules. For the other pairs, first assume $C_S(P) \xrightarrow{a} R$. There are two possible cases to consider:

1. $R = C_{S'}(P')$ with $S \xrightarrow{a}_{\Diamond} S'$ (or $S \xrightarrow{a}_{\Box} S'$), $C_S \xrightarrow[a]{a} C_{S'}$ and $P \xrightarrow{a} P'$,
2. $R = D_{S'}(P)$ with $S \xrightarrow{a}_{\Diamond} S'$, $S \not\xrightarrow{a}_{\Box} S'$, $C_S \xrightarrow[0]{a} D_{S'}$.

Note, that transitions $C_S(P) \xrightarrow{x}$ using transductions of the form $C_S \xrightarrow[a]{x}$ (where $S \not\xrightarrow{a}_{\Diamond}$) are not possible as $P \lhd S$ is assumed. We now construct matching transitions for Q_S as follows:

1. As S is *deterministic*, $S \xrightarrow{a}_{\Diamond} S'$ must be the transition that matches $P \xrightarrow{a} P'$. Hence $P' \lhd S'$, whence $Q_S \xrightarrow{a} Q_{S'}$ (because $S \xrightarrow{a}_{\Diamond} S'$) will be a matching transition.
2. As $S \xrightarrow{a}_{\Diamond} S'$, $Q_S \xrightarrow{a} Q_{S'}$ will be a matching transition.

Now, assume $Q_S \xrightarrow{a} Q_{S'}$. Then $S \xrightarrow{a}_{\Diamond} S'$. If $S \not\xrightarrow{a}_{\Box} S'$, $C_S(P) \xrightarrow{a} D_{S'}(P)$ will be a matching move. Otherwise $S \xrightarrow{a}_{\Box} S'$, whence $P \xrightarrow{a} P'$ with $P' \lhd S'$ for some P'. Also, $C_S \xrightarrow[a]{a} C_{S'}$. Hence, $C_S(P) \xrightarrow{a} C_{S'}(P')$ will be a matching transition.

For the $\Leftarrow$–direction, we show that $\mathcal{R} = \{(P, S) \,|\, C_S(P) \sim Q_S\}$ is a refinement (we consider only processes P with sort$(P) \subseteq$ sort(S)). Assume $P \xrightarrow{a} P'$. First note that $S \not\xrightarrow{a}_{\Diamond}$ is impossible as this would mean that $C_S(P) \xrightarrow{x}$, but (as x is not in the sort of S) $Q_S \not\xrightarrow{x}$. Thus, $S \xrightarrow{a}_{\Diamond} S'$ for some S' and hence $C_S \xrightarrow[a]{a} C_{S'}$. Thus $C_S(P) \xrightarrow{a} C_{S'}(P')$. Since S is deterministic, there is only one a-transition from S. Hence $Q_S \xrightarrow{a} Q_{S'}$ must be the matching move (in $C_S(P) \sim Q_S$) for which $C_{S'}(P') \sim Q_{S'}$ and hence $(P', S') \in \mathcal{R}$.

Now, assume $S \xrightarrow{a}_{\Box} S'$. We must find a matching move for P. But then $Q_S \xrightarrow{a} Q_{S'}$ (as $\xrightarrow{a}_{\Box} \subseteq \xrightarrow{a}_{\Diamond}$). Since $C_S(P) \sim Q_S$ it follows that $C_S(P) \xrightarrow{a} C_{S'}(P')$ for some P' with $P \xrightarrow{a} P'$ and $C_{S'}(P') \sim Q_{S'}$ (as $C_S \xrightarrow[a]{a} C'_S$ is the only a–producing transduction of C_S). Clearly $P \xrightarrow{a} P'$ is a matching move. □

Theorem 4.2 *The decision problem* EQ *is PSPACE–hard.*

Proof: Let $\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$ be an instance of INEQ_d . Then $\mathcal{E}^* = \{(C_{S_i} \circ C_i, Q_{S_i}) \mid i \in I\}$ is an instance of EQ and it follows from Lemma 4.1 that the solutionsets to $\mathcal{E}$ and $\mathcal{E}^*$ coincide, and that the size of $\mathcal{E}^*$ is polynomial in the size of $\mathcal{E}$. □

Now, let EQ^1 be the subproblem of EQ where only singleton collections are allowed. Then PSPACE–hardness follows from the PSPACE–hardness of INEQ^1 . Also the subproblems EQ_d and EQ_d^1 of EQ and EQ^1 , where only deterministic processes are allowed, are PSPACE–hard as the process Q_S constructed in Lemma 4.1 is deterministic provided S is.

5 Equation Solving in Process Algebra

In this section, we first consider the problem CCSEQ , which is the subproblem of EQ^1 obtained by restricting the context C to be of the form $(A \mid P)\backslash L$ for given A and L. We thereafter consider CCSOBS which is obtained from CCSEQ by replacing bisimulation equivalence $\sim$ by observation equivalence $\approx$.

The problem CCSEQ is the following:

> **Instance:** Finite-state processes A and B and a finite set of actions L.
>
> **Question:** Does there exist a process P such that the equation $(A \mid P)\backslash L \sim B$ is satisfied?

The size of an instance is the sum of the number of states and transitions in A and B and the number of actions in L. Let CCSEQ_d be the subproblem of CCSEQ where the process B is required to be deterministic. We shall prove that CCSEQ_d (and hence also CCSEQ) is PSPACE–hard by a reduction from the problem INEQ_R^1 , presented in Section 3. The following lemma provides the basis for this result. We extend the definition of rigid to modal specifications and contexts by saying that a modal specification (context) is rigid if none of its derivatives can perform a transition (transduction) labelled by (consuming or producing) the silent action τ.

Lemma 5.1 *Let C be a context and S be a modal specification such that S is deterministic, C and S are rigid, and (C, S) is contained in some correspondence relation $\propto$. Let L be the union of the sorts of C and S. Then there are stable processes A_C and B_S, such that for any rigid process P whose sort is contained in L*

$$C(P) \lhd S \quad \Leftrightarrow \quad (A_C \mid P)\backslash L \sim B_S \quad .$$

Moreover, the size of both A_C and B_S is linear in the size of C and S, and B_S is deterministic.

Note that a deterministic process is allowed to have τ-transitions (as long as there is at most one from each derivative).

Proof: The sorts of A_C and B_S is the union of L and L', where $L' = \{a' \mid a \in L\}$ is a tagged copy of L. For each context C and action $b \in L$ we define processes A_C and A_{Cb}. For each specification S and action $b \in L$ we define processes B_S and B_{Sb}. We state the inference rules defining the transitions of A_C, A_{Cb}, B_S, and B_{Sb} in terms of the transductions of C and the transitions of S. In these rules, a and b range over L, and a_0 ranges over $L \cup \{0\}$, and w is a distinguished action not in L.

$$\frac{C \xrightarrow[a]{b} C'}{A_C \xrightarrow{b'} A_{Cb} \quad A_{Cb} \xrightarrow{\overline{a}} A_{C'}} \qquad \frac{C \xrightarrow[0]{b} C'}{A_C \xrightarrow{b'} A_{Cb} \quad A_{Cb} \xrightarrow{\tau} A_{C'}} \qquad \frac{S \xrightarrow{b}_{\diamond} S' \quad S \not\xrightarrow{b}_{\Box} S' \quad (C,S) \in \propto}{A_C \xrightarrow{b'} A_{Cb} \quad A_{Cb} \xrightarrow{\tau} B_{S'}}$$

$$\frac{C \xrightarrow[a_0]{b} C' \quad (C,S) \in \propto}{B_S \xrightarrow{b'} B_{Sb}} \qquad \frac{S \xrightarrow{b}_{\diamond} S'}{B_S \xrightarrow{b'} B_{Sb} \quad B_{Sb} \xrightarrow{\tau} B_{S'}} \qquad A_C \xrightarrow{w} A_C \quad B_S \xrightarrow{w} B_S$$

Note that B_S is deterministic if S is deterministic. The idea is to let each transition of $C(P)$ and of S be imitated by two transitions, the first one with the same action and the second one with a silent action. During the silent action, the processes A_{Cb} can imitate the consumption of an action from the inner process of $C(P)$ by communicating with P. Define the relation $\mathcal{R}$ by $\mathcal{R} = \{(C(P), S) \mid C(P) \lhd S \quad \text{and} \quad (C,S) \in \propto\}$. By Lemma 3.4 $\mathcal{R}$ is a refinement.

For the $\Rightarrow$–direction, we show that

$$\begin{aligned} \mathcal{B} \;=\; & \{(A_C \mid P)\backslash L, B_S) \mid (C(P), S) \in \mathcal{R}\} \\ \cup\; & \{(A_{Cb} \mid P)\backslash L, B_{Sb}) \mid (C(P), S) \in \mathcal{R}\} \\ \cup\; & \{(B_S \mid P)\backslash L, B_S)\} \\ \cup\; & \{(B_{Sb} \mid P)\backslash L, B_{Sb})\} \end{aligned}$$

is a bisimulation if P is rigid and $\mathrm{sort}(P) \subseteq L$. For the pairs on the two bottom lines, this is obvious since the behavior of $B_S \mid P)\backslash L$ is identical to that of B_S. For the other pairs, first assume $(A_C \mid P)\backslash L \xrightarrow{b'} R$. Since b' is not in the sort of P we have $R = (A_{Cb} \mid P)\backslash L$ and either $C \xrightarrow[a_0]{b} C'$ or $S \xrightarrow{b}_{\diamond} S'$, and thus there is a move of form $B_S \xrightarrow{b'} B_{Sb}$. Next assume $(A_{Cb} \mid P)\backslash L \xrightarrow{\tau} R$. Since P is rigid, there are two cases to consider.

1. $R = (A_{C'} \mid P')\backslash L$ for some C', P', and a_0 with $C \xrightarrow[a_0]{b} C'$ and $P \xrightarrow{a_0} P'$, from which it follows that $C(P) \xrightarrow{b} C'(P')$. Since $(C(P), S) \in \mathcal{R}$ there is an S' with $S \xrightarrow{b}_{\diamond} S'$ and $(C'(P'), S') \in \mathcal{R}$. Thus, $B_{Sb} \xrightarrow{\tau} B_{S'}$ is a simulating move.
2. $R = (B_{S'} \mid P)\backslash L$ where $(C,S) \in \propto$. Since S is the only specification related to C by $\propto$ (due to the definition of correspondence relations) $S \xrightarrow{b}_{\diamond} S'$ is the only transition which can give rise to $(A_{Cb} \mid P)\backslash L \xrightarrow{\tau} R$. Thus we have $S \xrightarrow{b}_{\diamond} S'$. Thus $B_{Sb} \xrightarrow{\tau} B_{S'}$ is a simulating move.

Now assume $B_S \xrightarrow{b'} B_{Sb}$. There are three cases to consider:

1. $S \xrightarrow{b}_{\Box} S'$ for some S'. Since $C(P) \lhd S$, there must be a_0 and C' such that $C \xrightarrow[a_0]{b} C'$. Thus $(A_C \mid P)\backslash L \xrightarrow{b'} (A_{Cb} \mid P)\backslash L$ is a simulating move.
2. $S \xrightarrow{b}_{\diamond} S'$ and $S \not\xrightarrow{b}_{\Box} S'$ for some S'. By the inference rules, since $(C,S) \in \propto$, we have that $(A_C \mid P)\backslash L \xrightarrow{b'} (A_{Cb} \mid P)\backslash L$ is a simulating move.
3. $C \xrightarrow[a_0]{b} C'$ for some a_0. This goes back to the first case.

Finally assume $B_{Sb} \xrightarrow{\tau} B_{S'}$ for some S'. This implies that $S \xrightarrow{b}_{\diamond} S'$. In the case that $S \not\xrightarrow{b}_{\Box} S'$ and $(C,S) \in \propto$, this move is simulated by $(A_{Cb} \mid P)\backslash L \xrightarrow{\tau} (A'_{S'} \mid P)\backslash L$. In the case that $S \xrightarrow{b}_{\Box} S'$, there must be a_0, C' and P' such that $C \xrightarrow[a_0]{b} C'$ and $P \xrightarrow{a_0} P'$ and $(C'(P'), S') \in \mathcal{R}$. Thus $(A_{Cb} \mid P)\backslash L \xrightarrow{\tau} (A_{C'} \mid P')\backslash L$ is a simulating move.

For the $\Leftarrow$–direction, we show that

$$\mathcal{R} = \{(C(P), S) \mid (A_C \mid P)\backslash L \sim B_S\}$$

is a refinement. Assume $P \xrightarrow{a_0} P'$ and $C \xrightarrow[a_0]{b} C'$. This means that either $A_C \xrightarrow{b'} A_{Cb} \xrightarrow{\overline{a}} A_{C'}$ or $A_C \xrightarrow{b'} A_{Cb} \xrightarrow{\tau} A_{C'}$, i.e., that $(A_C \mid P)\backslash L \xrightarrow{b'} (A_{Cb} \mid P)\backslash L \xrightarrow{\tau} (A_{C'} \mid P')$. Since we have $(A_C \mid P)\backslash L \sim B_S$ we must have $B_S \xrightarrow{b'} B_{Sb} \xrightarrow{\tau} B_{S'}$ for some S' with $(A_{C'} \mid P')\backslash L \sim B_{S'}$ whence $S \xrightarrow{b}_{\diamond} S'$ is a matching move. Finally, assume that $S \xrightarrow{b}_{\Box} S'$. Thus we must have $B_S \xrightarrow{b'} B_{Sb} \xrightarrow{\tau} B_{S'}$ and consequently $(A_C \mid P)\backslash L \xrightarrow{b'} (A_{Cb} \mid P)\backslash L \xrightarrow{\tau} (A_{C'} \mid P')$ (since A_{Cb} can only communicate with P) with $(A_{C'} \mid P')\backslash L \sim B_{S'}$ for some a_0, C', P' with $P \xrightarrow{a_0} P'$ and $C \xrightarrow[a_0]{b} C'$. Thus $C(P) \xrightarrow{b} C'(P')$ is a matching move. □

To prove PSPACE-hardness of CcsEq$_d$, we must also get rid of the extra sort restriction imposed in Lemma 5.1. If A is a process and L is a set of actions, define the process A_L to have the same transitions as A and in addition the transition $A_L \xrightarrow{w} U$, where w is a distinguished action, and U (and U') is defined by the transitions

$$U \xrightarrow{\overline{a}} U' \qquad U \xrightarrow{a} U' \qquad U \xrightarrow{\tau} U' \qquad U' \xrightarrow{w} U$$

for each $a \in L$. If B is a process and L is a set of actions, define the process B^L to have the same transitions as B and in addition the transition $B^L \xrightarrow{w} V$, where w is a distinguished action and V is defined by the transitions $V \xrightarrow{\tau} V' \xrightarrow{w} V$.

Lemma 5.2 *Assume that A and B are stable processes and L is a set of actions. Then*

1. $\exists P' \,.\, (A_L \mid P')\backslash L \sim B^L$ *iff there is a rigid process P whose sort is in L such that* $(A \mid P)\backslash L \sim B$, *and*
2. $\exists P' \,.\, (A_L \mid P')\backslash L \approx B^L$ *iff there is a process P whose sort is in L such that* $(A \mid P)\backslash L \approx B$.

Moreover, the size of both A_L and B^L are linear in the size of A and B, and B^L is deterministic if B is deterministic.

Proof: We first prove claim 1. It is easy to see that a rigid process P with $\mathrm{sort}(P) \subseteq L$ which satisfies $(A \mid P)\backslash L \sim B$ also satisfies $(A_L \mid P)\backslash L \sim B^L$. To prove the claim in the reverse direction, if $(A_L \mid P)\backslash L \sim B^L$ then the transition $(A_L \mid P)\backslash L \xrightarrow{w} (U \mid P)\backslash L$ must be simulated by $B^L \xrightarrow{w} V$, and from $(U \mid P)\backslash L \sim V$ it follows that P is rigid and that $\mathrm{sort}(P) \subseteq L$. The second claim is proven analogously. Note that in this proof, we need to assume that A and B are stable in order to infer from $(A \mid P)\backslash L \approx B$ that $(A_L \mid P)\backslash L \approx B^L$ □

Theorem 5.3 *The decision problem* CcsEq$_d$ *is PSPACE–hard.*

Proof: Let CcsEq$_s$ be obtained by adding to CcsEq$_d$ the restrictions that the sought process P must be rigid and $\mathrm{sort}(P) \subseteq L$. By Lemma 5.1, InEq$^1_R$ can be reduced to CcsEq$_s$ (since if an instance C, S of InEq^1_R is solvable then there is a solution with a rigid process whose sort is in the union of the sorts of C and S). By the first part of Lemma 5.2, CcsEq$_s$ can be reduced to CcsEq$_d$. It follows by Theorem 3.3 that CcsEq$_d$ is PSPACE-hard. □

Next we consider the problem CcsObs , which is similar to CcsEq except that the satisfaction relation is that of observation equivalence, sometimes called weak bisimularity, denoted $\approx$. That is the problem is:

Instance: Finite-state processes A and B and a finite set of actions L.

Question: Does there exist a process P such that the equation $(A \mid P)\backslash L \approx B$ is satisfied?

Let CCSOBS$_d$ be the subproblem of CCSOBS where the process B is required to be deterministic and rigid. We shall prove that CCSOBS$_d$ (and hence also CCSOBS) is PSPACE–hard by a reduction from the problem CCSEQ$_d$. The following lemma provides the basis for this result.

Define the *rigidification* P_r of a process P as the process which has behaviour as follows: $P_r \xrightarrow{a} Q$ if and only if $P \stackrel{a}{\Longrightarrow} P'$ and $Q = P'_r$ where $a \neq \tau$. Note that P_r is rigid.

Lemma 5.4 *Let A_C, B_S and L be as in the proof of Lemma 5.1. Then for any process P with $sort(P) \subseteq L$ the following holds:*

$$(A_C \mid P)\backslash L \approx B_S \;\Rightarrow\; (A_C \mid P_r)\backslash L \sim B_S$$

Proof: We shall prove that

$$\begin{aligned}\mathcal{B} \;=\;& \{((A_C \mid P_r)\backslash L, B_S) \mid (A_C \mid P)\backslash L \approx B_S\} \\ \cup\;& \{((A_{Cb} \mid P_r)\backslash L, B_{Sb}) \mid (A_C \mid P)\backslash L \approx B_S\} \\ \cup\;& \{((B_{S'} \mid P_r)\backslash L, B_S) \mid (B_{S'} \mid P)\backslash L \approx B_S\} \\ \cup\;& \{((B_{S'b} \mid P_r)\backslash L, B_{Sb}) \mid (B_{S'} \mid P)\backslash L \approx B_S\}\end{aligned}$$

is a bisimulation. For the pairs on the two bottom lines, this follows from the observation that $B_S \approx (B_S)_r$ and $(B_{S'} \mid P)\backslash L \approx B_{S'}$, which if $(B_{S'} \mid P)\backslash L \approx B_S$ implies $(B_S)_r \approx (B_{S'})_r$ and hence also $(B_S)_r \sim (B_{S'})_r$. By inspecting processes of form B_S we infer that $B_S \sim B_{S'}$, whence $B_S \sim B_{S'} \sim (B_{S'} \mid P)\backslash L$ gives the pairs on the third line. The last line is proven similarly. For the pairs on the first line, $(A_C \mid P)\backslash L \approx B_S$ implies that $(A_C \mid P_r)\backslash L \xrightarrow{b'} (A_{Cb} \mid P_r)\backslash L$ iff $B_S \xrightarrow{b'} B_{Sb}$. For the pairs on the second line, note that $(A_{Cb} \mid P_r)\backslash L \stackrel{w}{\Longrightarrow}$ iff $B_{Sb} \stackrel{w}{\Longrightarrow}$. Thus there are two cases to consider. If $(A_{Cb} \mid P_r)\backslash L \stackrel{w}{\not\Longrightarrow}$ and $B_{Sb} \stackrel{w}{\not\Longrightarrow}$, we find that both $(A_{Cb} \mid P_r)\backslash L$ and B_{Sb} are deadlocked. If $(A_{Cb} \mid P_r)\backslash L \stackrel{w}{\Longrightarrow}$ and $B_{Sb} \stackrel{w}{\Longrightarrow}$, we find that $(A_{Cb} \mid P_r)\backslash L \xrightarrow{\tau} (R \mid P'_r)\backslash L$ for some R (which is either of form $A_{C'}$ or $B_{S''}$) and P'_r if and only if $B_{Sb} \xrightarrow{\tau} B_{S'}$ for some S'. Since $B_{Sb} \xrightarrow{\tau} B_{S'}$ is the only transition from B_{Sb} (recall that S is determinstic) we must have $(R \mid P'_r)\backslash L \approx B_{S'}$, which proves the lemma. □

Theorem 5.5 *The decision problem* CCSOBS$_d$ *is PSPACE–hard.*

Proof: Let A_C, B_S and L be as in the proof of Lemma 5.1. As $\approx$ is a weaker equivalence than $\sim$ it follows that

$$\exists P.\,((A_C)_L \mid P)\backslash L \sim (B_S)^L \implies \exists P.\,((A_C)_L \mid P)\backslash L \approx (B_S)^L$$

The opposite implication follows by combining Lemma 5.4 and Lemma 5.2 as follows.

$$\begin{aligned}& \exists P.\,((A_C)_L \mid P)\backslash L \approx (B_S)^L \\ \Longrightarrow\;& \exists P.\,[\mathrm{sort}(P) \subseteq L \;\text{ and }\; (A_C \mid P)\backslash L \approx B_S] \\ \Longrightarrow\;& \exists P.\,[P \text{ rigid} \;\text{ and }\; \mathrm{sort}(P) \subseteq L \;\text{ and }\; (A_C \mid P)\backslash L \sim B_S] \\ \Longrightarrow\;& \exists P.\,((A_C)_L \mid P)\backslash L \sim (B_S)^L\end{aligned}$$

Thus the two sides in the implication are equivalent. By Theorem 5.3 any instance (C, S) of INEQ^1_R can be reduced to an instance $((A_C)_L, (B_S)^L, L)$ of CCSEQ$_d$. We have just shown that this is also an equivalent instance of CCSOBS$_d$, whence the theorem follows from Theorem 3.3. □

6 Bounding the Number of Actions

In the equation solving problems considered in this paper, no restrictions have been placed on the set of actions of the involved processes and modal specifications. In some applications, it would be natural to let the number of actions be fixed when the size of the process grows. For instance, if the communication structure between the components in the process $(A \mid P)\backslash L \sim B$ is fixed, then the sorts of the involved components are restricted by the sorts induced by the communication structure. In this section, we shall prove that the problems considered in this paper are PSPACE-hard even when the number of actions is bounded by a small constant. The proof consists in showing that we can reduce a certain subproblem of InEq to a subproblem of InEq with only two different actions.

Let (C, S) be an instance of InEq[1] in which the sort of S and the outer sort of C contain only the actions x and y. Assume that the number of actions in the inner sort of C is at most 2^n. We can then find an encoding of each action a in the inner sort of C as a unique string $encode(a)$ of length n containing only the actions x and y. Define $encode(0)$ as a sequence of n 0's. We let σ range over strings of the actions x and y and over strings of 0's, and let ε denote the empty string. For each context C and each string σ which is a proper prefix of some $encode(a_0)$ with $C \xrightarrow[a_0]{u} C'$ for some u, we define a context C_σ. Here and in the following, a_0 ranges over $A \cup \{0\}$, u and v range over x and y, and v_0 ranges over $\{x, y, 0\}$. The transductions of these contexts are defined as follows.

$$\frac{C \xrightarrow[a_0]{u} C' \qquad \sigma v_0 \prec encode(a_0)}{C_\sigma \xrightarrow[v_0]{x} C_{\sigma v_0}} \qquad \frac{C \xrightarrow[a_0]{u} C' \qquad \sigma v_0 = encode(a_0)}{C_\sigma \xrightarrow[v_0]{u} C'_\varepsilon}$$

Here $\sigma \prec \sigma'$ denotes that the string σ is a proper prefix of the string σ'. The idea of the above transformation is that the transduction $C \xrightarrow[a_0]{u} C'$ is transformed into a sequence of n transductions from C_ε in which the sequence $encode(a_0)$ is consumed from the inner process, and a sequence with $n-1$ x's ending in u is produced. The $n-1$ first transductions from C_ε produce only x without giving any information.

For each process P and each string σ of x's and y's which is a proper prefix of some $encode(a)$ with $P \xrightarrow{a} P'$ for some P', we define a process P_σ. The transitions of these processes are defined as follows.

$$\frac{P \xrightarrow{a} P' \qquad \sigma v \prec encode(a)}{P_\sigma \xrightarrow{v} P_{\sigma v}} \qquad \frac{P \xrightarrow{a} P' \qquad \sigma v = encode(a)}{P_\sigma \xrightarrow{v} P'_\varepsilon}$$

The idea of the above transformation is that the transition $P \xrightarrow{a} P'$ is transformed into a sequence of n transitions from P_ε in which the sequence $encode(a)$ is produced. By convention, we shall identify P_σ with P when σ is a sequence of 0's. We shall also need an inverse transformation on processes, defined as follows. For each process P define the process $\tilde{P}$ by the following transitions:

$$\frac{P \xrightarrow{v_1} P_1 \xrightarrow{v_2} \cdots \xrightarrow{v_n} P' \qquad encode(a) = v_1 v_2 \ldots v_n}{\tilde{P} \xrightarrow{a} \tilde{P}'}$$

For each modal specification S and each integer i with $0 \leq i < n$ we define a modal specification S_i. The transitions between these are derived as follows:

$$\frac{S \not\xrightarrow{x}_\Box \qquad S \not\xrightarrow{y}_\Box \qquad 0 \leq i < n-1}{S_i \xrightarrow{x}_\Diamond S_{i+1}} \qquad \frac{S \xrightarrow{u}_\Diamond S'}{S_{n-1} \xrightarrow{u}_\Diamond S'_0}$$

$$\frac{S \xrightarrow{u}_{\Box} S' \qquad 0 \leq i < n-1}{S_i \xrightarrow{x}_{\Box} S_{i+1}} \qquad \frac{S \xrightarrow{u}_{\Box} S'}{S_{n-1} \xrightarrow{u}_{\Box} S'_0}$$

Lemma 6.1 *If S is deterministic, then S_0 is deterministic. Furthermore, if (C, S) are included in some corespondence relation, then there is a correspondence relation which includes (C_ε, S_0).*

Proof: To see that S_0 is deterministic if S is, note that $S_i \xrightarrow{x}_{\Diamond} S_{i+1}$ is the only transition from S_i when $i < n-1$, and that $S_{n-1} \xrightarrow{u}_{\Diamond} S'_0$ implies $S \xrightarrow{u}_{\Diamond} S'$. For the second claim, assume that there is a correspondence relation $\propto$ which includes (C, S). Define the correspondence relation $\propto'$ by

$$\propto' = \{(C_\sigma, S_i) \mid (C, S) \in \propto \qquad \text{and} \qquad i = length(\sigma)\}$$

It is easy to see that $\propto'$ is a correspondence relation if $\propto$ is. □

Lemma 6.2 *Let C be a context, P be a process, and S be a modal specification such that the sort of S and the outer sort of C contain only x and y, and (C, S) is contained in a safe correspondence relation $\propto$. If $C(P) \lhd S$ then $C_\varepsilon(P_\varepsilon) \lhd S_0$.*

Proof: We shall prove that

$$\mathcal{R} = \{(C_\sigma(P_\sigma), S_i) \mid C(P) \lhd S \text{ and } (C, S) \in \propto \text{ and } i = length(\sigma)\}$$

is a refinement. It is easy to see that $C_\sigma(P_\sigma) \xrightarrow{x} C_{\sigma v_0}(P_{\sigma v_0})$ is simulated by $S_i \xrightarrow{x}_{\Diamond} S_{i+1}$ whenever $i < n-1$. If there is a transition of form $C_\sigma(P_\sigma) \xrightarrow{u} C'_\varepsilon(P'_\varepsilon)$, then $C_\sigma \xrightarrow[v_0]{u} C'_\varepsilon$ and $P_\sigma \xrightarrow{v_0} P'_\varepsilon$ for some v_0. It follows that $C \xrightarrow[a_0]{u} C'$ and $P \xrightarrow{a_0} P'$ where $\sigma v_0 = encode(a_0)$. Thus $C(P) \xrightarrow{u} C'(P')$, whence there must be a S' with $S \xrightarrow{u}_{\Diamond} S'$ and $C'(P') \lhd S'$ and $(C', S') \in \propto$. Hence there is a transition of form $S_{n-1} \xrightarrow{u}_{\Diamond} S'_0$. By definition we have $(C'_\varepsilon(P'_\varepsilon), S'_0) \in \mathcal{R}$.

Next, we check $S_i \xrightarrow{x}_{\Box} S_{i+1}$ for $i < n-1$. This implies that $S \xrightarrow{u}_{\Box} S'$ for some u and S'. Hence $C(P) \xrightarrow{u} C'(P')$ for some C', P' with $C'(P') \lhd S'$ and $(C', S') \in \propto$ whence $C \xrightarrow[a_0]{u} C'$ and $P \xrightarrow{a_0} P'$ for some a_0. Since $\propto$ is safe, we now have two cases to consider.

1. $C \xrightarrow[a_0]{u} C'$ is the only transduction from C. In this case the existence of C_σ implies that $\sigma \prec encode(a_0)$. Thus for v_0 with $\sigma v_0 \prec encode(a_0)$ we have $C_\sigma \xrightarrow[v_0]{x} C_{\sigma v_0}$ and $P_\sigma \xrightarrow{v_0} P_{\sigma v_0}$, Hence $C_\sigma(P_\sigma) \xrightarrow{x} C_{\sigma v_0}(P_{\sigma v_0})$.

2. For all a there is a C' with $C \xrightarrow[a]{u} C'$. Hence there is both the transduction $C_\sigma \xrightarrow[x]{x} C_{\sigma x}$ and/or $C_\sigma \xrightarrow[y]{x} C_{\sigma y}$ whenever σx and/or σy are prefixes of some $encode(a)$. The definition of P_σ implies that for some v we have $P_\sigma \xrightarrow{v} P_{\sigma v}$. Regardless of whether v is x or y in $P_\sigma \xrightarrow{v} P_{\sigma v}$, there is a transduction $C_\sigma \xrightarrow[v]{x} C_{\sigma v}$ which implies that $C_\sigma(P_\sigma) \xrightarrow{x} C_{\sigma v}(P_{\sigma v})$. In the case that σ is a sequence of 0's, it is easy to see that $C_\sigma(P_\sigma) \xrightarrow{x} C_{\sigma v}(P_{\sigma v})$.

Finally, we check the case $S_{n-1} \xrightarrow{u}_{\Box} S'_0$. We get the same two cases as for the case $S_i \xrightarrow{x}_{\Box} S_{i+1}$. In case 1 we find that for v_0 with $\sigma v_0 = encode(a_0)$ we have $C_\sigma \xrightarrow[v_0]{u} C'_\varepsilon$ and $P_\sigma \xrightarrow{v_0} P'_\varepsilon$, Hence

$C_\sigma(P_\sigma) \xrightarrow{u} C'_\varepsilon(P'_\varepsilon)$. In case 2 we find that regardless of whether v is x or y in $P_\sigma \xrightarrow{v} P'_\varepsilon$ there is a matching transduction $C_\sigma \xrightarrow[v]{u'} C'_\varepsilon$ which implies that $C_\sigma(P_\sigma) \xrightarrow{u'} C'_\varepsilon(P'_\varepsilon)$. Thus there is a transition $C(P) \xrightarrow{u'} C'(P')$ which must be matched by $S \xrightarrow{u'}_\diamond S''$ for some S''. However, by the definition of safe correspondence relations there is only one transition from S whence $u = u'$ and $S' = S''$. □

Next we shall prove a lemma in the reverse direction.

Lemma 6.3 *Let C be a context, P be a process, and S be a modal specification where the sort of S and the outer sort of C contain only x and y. If $C_\varepsilon(P) \lhd S_0$ then $C(\tilde{P}) \lhd S$.*

Proof: Assume that $C_\varepsilon(P) \lhd S_0$ and that $C(\tilde{P}) \xrightarrow{u} C'(\tilde{P}')$. This implies that there is a sequence of transitions $C_\varepsilon(P) \xrightarrow{x} C_{\sigma 1}(P_1) \xrightarrow{x} C_{\sigma 2}(P_2) \xrightarrow{x} \cdots C_{\sigma n-1}(P_{n-1}) \xrightarrow{u} C'_\varepsilon(P')$. Since $C_\varepsilon(P) \lhd S_0$ there is a sequence of transitions $S_0 \xrightarrow{x}_\diamond S_1 \xrightarrow{x}_\diamond S_2 \xrightarrow{x}_\diamond \cdots S_{n-1} \xrightarrow{u}_\diamond S'_0$ with $C'_\varepsilon(P') \lhd S'_0$ which implies that $S \xrightarrow{u}_\diamond S'$ and $C'(\tilde{P}') \lhd S'$. In an analogous manner, we conclude that if $S \xrightarrow{u}_\Box S'$. then there are C', $\tilde{P}'$ such that $C(\tilde{P}) \xrightarrow{u} C'(\tilde{P}')$ and $C'(\tilde{P}') \lhd S'$. □

Let INEQ_b be the subproblem of INEQ_d where in each instance $\{(C_i, S_i) \mid i \in I\}$ we require that there is a set of two elements which contains the outer sort of C_i and the sort of S_i for each $i \in I$. Let INEQ_{bR} be the subproblem of INEQ_b obtained by also requiring that (C_i, S_i) be contained in a safe correspondence relation. Let $\text{IN}\text{EQ}^1_{bR}$ be the subproblem of INEQ_{bR} where only singleton collections are allowed.

Theorem 6.4 *The problems INEQ_b , INEQ_{bR} , and $\text{IN}\text{EQ}^1_{bR}$ are PSPACE-hard.*

Proof: Follows from Lemmas 6.2 and 6.3 and Theorem 3.3. □

Let EQ_b be the subproblem EQ_d where in each instance $\{(C_i, P_i) \mid i \in I\}$ we require that the sorts of all C_i and P_i be contained in a set of three elements. Let CCSEQ_b be the subproblem CCSEQ_d where in each instance (A, B, L) we require that L and the sorts of A and B be contained in a set of six elements. Define CCSOBS_b analogously.

Theorem 6.5 *The problems EQ_b , CCSEQ_b , and CCSOBS_b are PSPACE-hard.*

Proof: Follows from Theorem 6.4 using the same reductions as in the proofs of the theorems 4.2, 5.3, and 5.5. □

7 Polynomial Equation Solving

As argued in the Introduction both equation and inequation solving occur during (top–down) development of concurrent systems. As such it is important to find conditions under which these problems may be dealt with efficiently. In this section we identify conditions on contexts which will induce a subproblem of EQ^1 which is solvable in polynomial time.

Definition 7.1 *A context C is* deterministic *if $D \xrightarrow[a]{b_1} D_1$ and $D \xrightarrow[a]{b_2} D_2$ implies $b_1 = b_2$ and $D_1 = D_2$ for any derivative D of C.*

Definition 7.2 *Let $\mathcal{P} = (S, A, \longrightarrow)$ be a labelled transition system and let $\mathcal{C} = (K, A, \longrightarrow)$ be a context system. A* consistency *relation $\mathcal{K}$ is a subset of $K \times S$ such that whenever $(C, Q) \in \mathcal{K}$ then the following holds:*

$$\textit{Whenever } Q \xrightarrow{b} Q', \textit{ then } C \xrightarrow[a]{b} C' \textit{ for some } a, C' \textit{ such that } (C', Q') \in \mathcal{K}.$$

We say that C and P are consistent if (C, P) is contained in some consistency relation.

Note that the notion of consistency is very similar to that of simulation (being "half" of bisimulation [Mil89]) for which there is a well–known polynomial time decision procedure [KS83]. For deterministic contexts the notion of consistency captures exactly that of solvability:

Theorem 7.3 *Let C be a deterministic context and Q a process. Then there exists a process P such that $C(P) \sim Q$ if and only if C and Q are consistent.*

Proof:
$\Rightarrow$ Let $\mathcal{K} = \{(C, Q) \mid \exists P. C(P) \sim Q\}$. We show that $\mathcal{K}$ is a consistency relation. So let $(C, Q) \in \mathcal{K}$ and let $Q \xrightarrow{b} Q'$. Now assume $C(P) \sim Q$, then $C(P) \xrightarrow{b} R$ with $R \sim Q'$. According to the rule of inference for combined processes, $C \xrightarrow[a]{b} C'$ and $P \xrightarrow{a} P'$ for some a, C' and P' with $R = C'(P')$. Obviously, $(C', Q') \in \mathcal{K}$.
$\Leftarrow$ Let $\mathcal{K}$ be a consistency relation. Define a transition system with states $P_{C,Q}$ for $(C, Q) \in \mathcal{K}$ and transtions:

$$P_{C,Q} \xrightarrow{a} P_{C',Q'} \quad \Leftrightarrow^{\Delta} \quad \exists b. Q \xrightarrow{b} Q' \wedge C \xrightarrow[a]{b} C'$$

Then $C(P_{C,Q}) \sim Q$ for all $(C, Q) \in \mathcal{K}$. To see this we show that the relation below is a bisimulation:

$$\mathcal{B} = \{(C(P_{C,Q}), Q) \mid (C, Q) \in \mathcal{K}\}$$

Let $(C(P_{C,Q}), Q) \in \mathcal{B}$ and assume $Q \xrightarrow{b} Q'$ As $(C, Q) \in \mathcal{K}$, $C \xrightarrow[a]{b} C'$ with $(C', Q') \in \mathcal{K}$ for some a, C'. But then $P_{C,Q} \xrightarrow{a} P_{C',Q'}$. Hence, using the inference rule for combined processes, $C(P_{C,Q}) \xrightarrow{b} C'(P_{C',Q'})$ and clearly $(C'(P_{C',Q'}), Q') \in \mathcal{B}$. Let $C(P_{C,Q}) \xrightarrow{b} R$. That is $C \xrightarrow[a]{b} C'$ and $P_{C,Q} \xrightarrow{a} P_{C'',Q''}$ for some a, C' and $P_{C'',Q''}$. Now according to the definition of $P_{C,Q}$'s transtions $C \xrightarrow[a]{b'} C''$ and $Q \xrightarrow{b'} Q''$ for some b'. But as C is assumed to be *deterministic* it follows that $b = b'$ and $C' = C''$. Thus $R = C'(P_{C',Q''})$ and obviously $Q \xrightarrow{b} Q''$ is a matching transition. □

Since consistency relations can be found in polynomial time, we get the following theorem.

Theorem 7.4 *Let C be a deterministic context and Q a process. Then the problem whether there exists a process P such that $C(P) \sim Q$ can be decided in polynomial time.*

Examples of deterministic contexts are:

- the CCS context $(A \,|\, [\,])\backslash L$, where A is deterministic and rigid (i.e. the derivatives of A have no internal transitions) and the sort of A is included in L,
- the CSP [BHR84] context $A|||[\,]$, where A is a deterministic process.

Open Problems and Future and Related Work

This paper leaves as open problems whether or not the decision problems considered in the paper are members of PSPACE or not. In [Lar91] it is shown that the problem INEQ , where the modal specification is allowed to be *nondeterministic*, is DEXPTIME-complete. This result, however, does not say anything about the complexity of the other problems, since their complexity bounds are obtained from the problem INEQ_d which requires a deterministic modal specification. The problems in the paper can all be solved in single exponential time, using a procedure which is easily derived from Larsen and Xinxin [LX90b]. This procedure involves checking for consistency in a (disjunctive) modal transition system with an *exponential size* in that of the underlying context and process.

On the positive side, a more careful examination shows that the procedure for EQ presented in [LX90b] has time complexity exponential in the size of the contexts but *polynomial* in the size of the processes. Thus, equation systems with small or bounded size contexts may be dealt with efficiently.

As for future work we should like to continue the work of Section 7 in identifying more liberal conditions on contexts which will make (in)equation solving efficient. Ongoing work in this direction is also carried out by Liu Xinxin.

The problem CCSOBS_d , i.e.,the equation $(A \mid P)\backslash L \approx B$ where B is required to be deterministic and rigid, has been studied by several authors. Shields [Shi89] presents a solution method for solving CCSOBS_d which yields an exponential time algorithm. Shields' formulation of the problem is that B is required to be observationally equivalent to a deterministic and rigid process. Since it is easy (polynomial time solvable) to find such an equivalent process, one sees that Shields essentially considers the problem CCSOBS_d . Qin and Lewis [QL90] also present an exponential time algorithm for CCSOBS_d . Their work extends and is based on the work by Shields. Parrow [Par89a] present a tableau transformation method for solving CCSOBS_d , and and implementation of the method as a semi-automatic program where the user can guide the search for a solution. The program can also run automatically and generate a most general solution, which however is often too large.

The equation $(A \mid P)\backslash L$ **sat** B where **sat** is trace equivalence has been treated by Merlin and Bochmann [MB83]. They define a method which yields a most general solution (allowing as many traces as possible). No results on the complexity of their method is given.

A related design problem is that of constructing a system P which satisfies a given formula in a temporal logic. For linear time temporal logic this problem has been considered by Manna and Wolper [MW84] obtaining a PSPACE-complete problem, and in a different framework by Pnueli and Rossner [PR89] obtaining a double exponential time algorithm. For branching time temporal logic (CTL) the problem has been considered by Clarke and Emerson [CE82].

Acknowledgment

The authors are grateful to Ed Brinksma, Joachim Parrow, and Pierre Wolper for fruitful discussions, and to the referees of CAAP '91 for helpful comments.

References

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BK85] J. Bergstra and J. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[BK86] J. Bergstra and J. Klop. Verification of an alternating bit protocol by means of proces algebra. In Bibel and Jantke, editors, *Mathematical methods of specification and synthesis of software systems '85*, volume 215 of *Lecture Notes in Computer Science*, pages 9–24. Springer Verlag, 1986.

[BL90] G. Boudol and Kim Larsen. Graphical versus logical specifications. In Arnold, editor, *Proc. Coll. on Trees and Algebra in Programming*, volume 431 of *Lecture Notes in Computer Science*, pages 57–71. Springer Verlag, 1990.

[Bou85] G. Boudol. Calcul de processus et verification. Technical Report 424, INRIA, 1985.

[CE82] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons using branching time temporal logic. *Science of Computer Programming*, 2:241–266, 1982.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-completeness.* W.H. Freeman and Company, New York, 1979.

[HL89] H. Hüttel and K.G. Larsen. The use of static constructs in a modal process logic. In Meyer and Taitslin, editors, *Proc. Logik at Botik '89*, volume 363 of *Lecture Notes in Computer Science*, pages 163 – 180. Springer Verlag, 1989.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[Koo85] C. Koomen. Algebraic specification and verification of protocols. *Science of Computer Programming*, 5(1):1–36, 1985.

[KS83] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *Proc. 2:nd ACM Symp. on Principles of Distributed Computing*, Montreal, Canada, 1983. To appear in *Information and Computation.*

[Lar89] K.G. Larsen. Modal specifications. In Sifakis, editor, *Proc. Int. Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer Verlag, 1989.

[Lar91] K.G. Larsen. The expressive power of implicit specifications, 1991. To appear in *Proc. ICALP '91.*

[LM87] K.G. Larsen and R. Milner. Verifying a protocol using relativized bisimulation. In Ottmann, editor, *Proc. ICALP '87*, volume 267 of *Lecture Notes in Computer Science*, pages 126–135. Springer Verlag, 1987.

[LT88] K.G. Larsen and B. Thomsen. A modal process logic. In *Proc.* 3^{rd} *IEEE Int. Symp. on Logic in Computer Science*, pages 203–210, 1988.

[LX90a] K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In *Proc. ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 526–539. Springer Verlag, 1990.

[LX90b] K.G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *Proc.* 5^{th} *IEEE Int. Symp. on Logic in Computer Science*, 1990.

[MB83] P. Merlin and G. von Bochmann. On the construction of submodule specifications and communication protocols. *ACM Trans. on Programming Languages and Systems*, 5(1):1–25, 1983.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes of Computer Science*. Springer Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[MW84] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 6(1):68–93, 1984.

[Par81] D. Park. Concurrency and automata on infinite sequences. In Deussen, editor, *Proc. 5^{th} GI Conf. of Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 245–251, 1981.

[Par89a] J. Parrow. Submodule construction as equation solving in CCS. *Theoretical Computer Science*, 68:175–202, 1989.

[Par89b] J. Parrow. Verifying a CSMA/CD-protocol with CCS. In *Protocol Specification, Testing, and Verification VIII*. North-Holland, 1989.

[Plo81] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Denmark, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In Brauer, editor, *Proc. ICALP '85*, volume 194 of *Lecture Notes in Computer Science*, pages 15–32. Springer Verlag, 1985.

[PR89] A. Pnueli and R. Rossner. On the synthesis of a reactive module. In *Proc. 16:th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[QL90] H. Qin and P. Lewis. Factorization of finite state machines under observational equivalence. In Baeten, editor, *Proc. CONCUR, Amsterdam*, volume 458 of *Lecture Notes in Computer Science*, pages 427–441. Springer Verlag, 1990.

[SFD85] S.A. Smolka, A.J. Frank, and S.K. Debray. Testing protocol robustness the CCS way. In *Protocol Specification, Testing, and Verification IV*, pages 93–108. North-Holland, 1985.

[Shi89] M.W. Shields. Implicit system specification and the interface equation. *The Computer Journal*, 32(5):399–412, 1989.

A Proof that INEQ is PSPACE–hard

A.1 The Generalized Geography Problem

Definition A.1 *A rooted, directed graph is a structure $G = (V, E, v_0)$, where V is a (finite) set of vertices, $E \subseteq V \times V$ is a set of edges and $v_0 \in V$ is the root (the initial vertex).*

Let $G = (V, E, v_0)$ be a rooted, directed graph. For $e = (u, v) \in E$ we write $\mathrm{hd}e$ for v and $\mathrm{tl}e$ for u. A *path* of G is any finite sequence $p = e_0 e_1 e_2 \ldots e_n$, where $\mathrm{tl}e_0 = v_0$ and $\mathrm{hd}e_i = \mathrm{tl}e_{i+1}$ for all $i \in [0, n[$. For $e \in E$ we define the set $\mathrm{Follow}(e) = \{f \in E \mid \mathrm{hd}e = \mathrm{tl}f\}$. Also, $\mathrm{Init} = \{e \in E \mid \mathrm{tl}e = v_0\}$. In fact, a sequence of edges $e_0 e_1 e_2 \ldots e_n$ is a path of G just in case $e_0 \in \mathrm{Init}$ and $e_{i+1} \in \mathrm{Follow}(e_i)$ for all $i \in [0, n[$.

Given a rooted, directed graph $G = (V, E, v_0)$ the (two–player) *Generalized Geography* game on G is played according to the following rules [GJ79]:

> The two players alternate choosing a new edge from E. The first edge chosen (by player 1) must have its tail at v_0 and each subsequently chosen edge must have its tail at the vertex that was the head of the previous edge, and must not have been previously chosen in the game. The first player unable to choose such a new edge loses.

Now, the Generalized Geography problem GENGEO may be described as below. Also, we recall from [GJ79] that GENGEO is PSPACE–complete.

Instance: A rooted, directed graph G.

Question: Does player 1 have a forced win in the Generalized Geography game played on G?

We want to reformulate (or formalize) the GENGEO problem into a question of existence of a process (of some labelled transition system) expressing in an explicit way a winning strategy for player 1 on a given graph G. Thus, let $G = (V, E, v_0)$ be a rooted, directed graph and let P be a process of some labelled transition system *with E as action set*. Then:

P respects G if $\mathrm{Act}(d)$ is a path of G for any finite derivation sequence d of P.

P obeys the GENGEO *game* if the actions (i.e. edges of G) occurring in any derivation sequence of P are all different.

The idea is that the computations of P should correspond to complete GENGEO games on G (with player 1 as winner if the length of the computation is odd). Now, we want P to capture several GENGEO games; in particular we want P to provide player 1 with a strategy for *any* legal move of the opponent:

P provides a strategy with respect to G if whenever

$$P \xrightarrow{e_0} P_1 \xrightarrow{e_1} P_2 \xrightarrow{e_2} \cdots \xrightarrow{e_j} P_j$$

is an odd length derivation sequence of P, then for any $e \in \mathrm{Follow}(e_j) \backslash \{e_0, \ldots, e_j\}$:

$$P_j \xrightarrow{e} P_{j+1}^e$$

for some P_{j+1}^e.

Here Follow$(e_j)\backslash\{e_0,\ldots,e_j\}$ is the set of legal moves of the opponent (only *new* edges can be chosen), and P^e_{j+1} describes player 1's strategy after the move e. Finally, P should only contain computations with player 1 as winner. I.e.:

> *P provides a winning strategy with respect to G* if P respects G, obeys the GENGEO game and provides a strategy wrt. G such that all computations of P have odd length.

We now reformulate (or formalize) the GENGEO problem as follows:

> **Instance:** A rooted, directed graph G.
>
> **Question:** Does there exist a process P providing a winning strategy with respect to G?

A.2 Inequation Solving

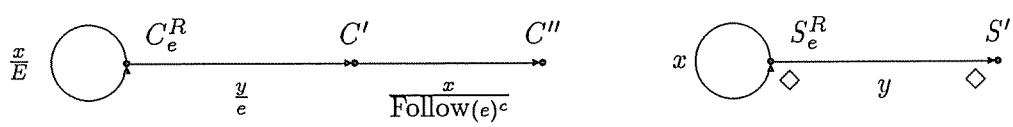
We recall that the problem of Inequation Solving INEQ is defined as follows:

> **Instance:** A finite collection of pairs
>
> $$\mathcal{E} = \{(C_i, S_i) \mid i \in I\}$$
>
> where I is a finite index set, and for all $i \in I$, C_i is a finite–state context and S_i is a finite–state modal specification.
>
> **Question:** Does there exist a process P such that the inequation $C_i(P) \lhd S_i$ is satisfied for all $i \in I$?

In the remainder of this section we shall show how to transform (in polynomial time) any instance $G = (V, E, v_0)$ of GENGEO into an equivalent instance $\mathcal{E}_G$ of INEQ . That is: there will be a winning strategy for player 1 on G just in case the inequation system $\mathcal{E}_G$ has a solution. In fact, the transformation offered will be such that the solutionset to $\mathcal{E}_G$ is exactly the set of processes that provide winning strategies with respect to G. As GENGEO is a PSPACE–complete decision problem it will follow that INEQ is PSPACE–hard. Whether or not INEQ is *in* PSPACE is as yet an open problem on which we shall comment in the conclusion.

In the remainder of this section let $G = (V, E, v_0)$ be a given rooted, directed graph. We construct, in the following lemmas, inequation systems which will be equivalent to the four conditions on a winning strategy for G.

Lemma A.2 *For $e \in E$ let C^R_e and S^R_e have the following behaviours* [2] *:*



[2] If, for $B_1, B_2 \subseteq E$, $\frac{B_2}{B_1}$ labels an edge between contexts C and D, this means by convention that $C \xrightarrow[a]{b} D$ for any $a \in B_1$ and $b \in B_2$. Singleton sets are identified with their element. For a set A, A^c denotes the complementary set.

where x and y are different actions. Also let C_I and S_I be defined by:

Then P is a solution to the inequation system:

$$\mathcal{E}_R = \{(C_e^R, S_e^R) \mid e \in E\} \cup \{(C_I, S_I)\}$$

if and only if P respects G.

Proof:
$\Leftarrow$ We show that the relation:

$$\begin{aligned}\mathcal{R} \;=\; & \{(C_e^R(Q), S_e^R) \mid e \in E,\ Q \text{ is a derivative of } P,\ P \text{ respects } G\} \\ & \cup\ \{(C_I(P), S_I) \mid P \text{ respects } G\}\end{aligned}$$

is a refinement up to $\lhd$ (i.e., that $\mathcal{R} \cup \lhd$ is a refinement). So consider $(C_e^R(Q), S_e^R) \in \mathcal{R}$ and consider the possible transitions of $C_e^R(Q)$. There are two cases to consider:

1. $C_e^R(Q) \xrightarrow{x} C_e^R(Q')$ with $Q \xrightarrow{f} Q'$ for some $f \in E$, or
2. $C_e^R(Q) \xrightarrow{y} C'(Q')$ with $Q \xrightarrow{e} Q'$.

In case 1, $S_e^R \xrightarrow{x}_\diamond S_e^R$ provides an obvious match with $(C_e^R(Q'), S_e^R) \in \mathcal{R}$. In case 2, we claim that $S_e^R \xrightarrow{y}_\diamond S'$ will provide a match as $C'(Q') \lhd S'$. Assume not. Then necessarily $C'(Q') \xrightarrow{x} C''(Q'')$ with $Q' \xrightarrow{f} Q''$ for some $f \notin \text{Follow}(e)$. However, this will contradict the assumption that Q was a derivative of a process P respecting G. Secondly, consider $(C_I(P), S_I) \in \mathcal{R}$ and consider the two types of transitions. Clearly, no transition $C_I(P) \xrightarrow{x}$ is possible as this would require $P \xrightarrow{e}$ for some $e \notin \text{Init}$, and thus contradicting the assumption that P respects G. Any transition $C_I(P) \xrightarrow{y}$ can obviously be matched by $S_I \xrightarrow{y}_\diamond$

$\Rightarrow$ Let P be a solution to $\mathcal{E}_R$. Assume P does not respect G. Then either $P \xrightarrow{e}$ for some $e \notin \text{Init}$ contradicting $C_I(P) \lhd S_I$ or P has a derivation sequence of the form:

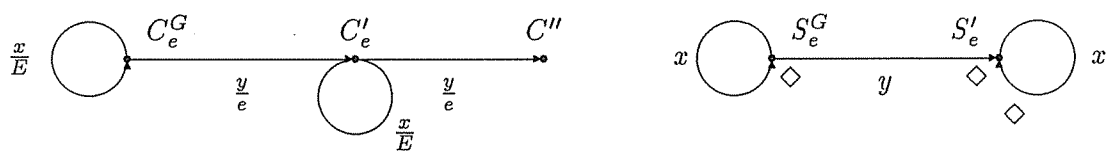
$$P \xrightarrow{e_1} P_1 \xrightarrow{e_2} \cdots P_{n-1} \xrightarrow{e_n} P_n \xrightarrow{e_{n+1}} P_{n+1}$$

where $e_{n+1} \notin \text{Follow}(e_n)$. However, this will contradict $C_{e_n}^R \lhd S_{e_n}^R$ as the transition sequence:

$$C_{e_n}^R(P) \xrightarrow{x} C_{e_n}^R(P_1) \xrightarrow{x} \cdots \xrightarrow{x} C_{e_n}^R(P_{n-1}) \xrightarrow{y} C'(P_n) \xrightarrow{x} C''(P_{n+1})$$

is not allowed by $S_{e_n}^R$. □

Lemma A.3 *For $e \in E$ let C_e^G and S_e^G have the following behaviours:*



where x, y are different actions. Then P is a solution to the inequation system:

$$\mathcal{E}_G = \{(C_e^G, S_e^G) \mid e \in E\}$$

if and only if P obeys the GenGeo *game.*

Proof:
$\Leftarrow$ Say that P *cannot* e if no derivative of P can perform the action e. We show that the relation:

$$\begin{aligned}\mathcal{R} \;=\; & \{(C_e^G(P), S_e^G) \mid e \in E,\ P \text{ obeys } G\} \\ & \cup\ \{(C'_e(P), S'_e) \mid e \in E,\ P \text{ cannot } e\}\end{aligned}$$

is a refinement up to $\lhd$. So consider $(C_e^G(P), S_e^G) \in \mathcal{R}$ and consider the possible transitions of $C_e^G(Q)$. There are two cases to consider:

1. $C_e^G(P) \xrightarrow{x} C_e^G(P')$ with $P \xrightarrow{f} P'$ for some $f \in E$, or
2. $C_e^G(P) \xrightarrow{y} C'_e(P')$ with $P \xrightarrow{e} P'$.

In case 1, $S_e^G \xrightarrow{x}_\diamond S_e^G$ provides an obvious match with $(C_e^G(P'), S_e^G) \in \mathcal{R}$. In case 2, $S_e^G \xrightarrow{y}_\diamond S'_e$ provides a match with $(C'_e(P'), S'_e) \in \mathcal{R}$ (since P obeys the GenGeo game we have P' cannot e). Next consider $(C'_e(P), S'_e) \in \mathcal{R}$. Since we assume that P cannot e there is only the transition $C'_e(P) \xrightarrow{x} C'_e(P')$ with $P \xrightarrow{f} P'$ for some $f \in E$. Here $S'_e \xrightarrow{x}_\diamond S'_e$ provides a match.

$\Rightarrow$ Let P be a solution to $\mathcal{E}_G$. Assume P does not obey G. Then P has a derivation sequence of the form:

$$P \xrightarrow{e_1} P_1 \xrightarrow{e_2} \cdots P_{m-1} \xrightarrow{e} P_m \xrightarrow{e_{m+1}} \cdots P_{n-1} \xrightarrow{e} P_n$$

in which the action e occurs twice (as e_m and as e_n). In this case we must have $C_e^G(P) \not\lhd S_e^G$, since the only possible match for:

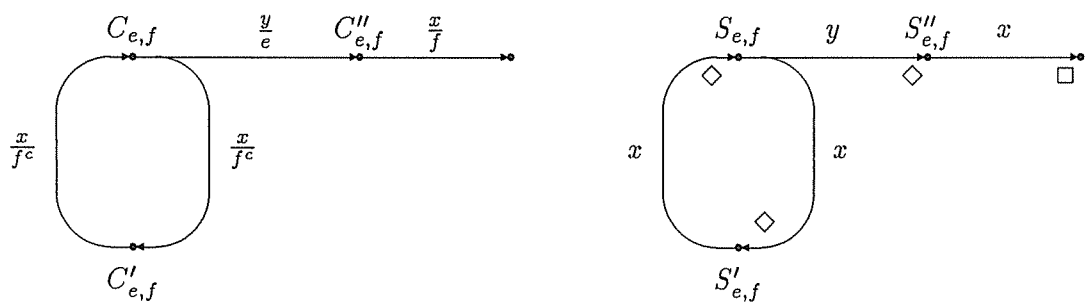
$$C_e^G(P) \xrightarrow{x} \cdots \xrightarrow{x} C_e^G(P_{m-1}) \xrightarrow{y} C'_e(P_m) \xrightarrow{x} \cdots \xrightarrow{x} C'_e(P_{n-1})$$

is

$$S_e^G \xrightarrow{x}_\diamond \cdots \xrightarrow{x}_\diamond S_e^G \xrightarrow{y}_\diamond S'_e \xrightarrow{x}_\diamond \cdots \xrightarrow{x}_\diamond S'_e$$

However, the transition $C'_e(P_{n-1}) \xrightarrow{y}$ as $P_{n-1} \xrightarrow{e}$ is not allowed by S'_e. □

Lemma A.4 *For $e \in E$ and $f \in \text{Follow}(e)\backslash\{e\}$ let $C_{e,f}$ and $S_{e,f}$ have the following behaviours:*



with x, y being different actions. Then P is a solution to the inequation system:

$$\mathcal{E}_S = \{(C_{e,f}, S_{e,f}) \mid e \in E, f \in \text{Follow}(e)\backslash\{e\}\}$$

if and only if P provides a strategy with respect to G.

Proof:
$\Leftarrow$ Consider the relation $\mathcal{R}$ consisting of all pairs $(C_{e,f}(P), S_{e,f})$, where Q is f^c–reachable in *even* length from some P providing a strategy for G, together with all pairs $(C'_{e,f}(P), S'_{e,f})$ where Q is f^c–reachable in *odd* length from some P providing a strategy for G. We will show that this relation $\mathcal{R}$ is a refinement (up to $\lhd$). Thus let $(C_{e,f}(P), S_{e,f}) \in \mathcal{R}$ and consider the various possible transitions for $C_{e,f}(P)$:

1. $C_{e,f}(P) \xrightarrow{x} C'_{e,f}(P')$ with $Q \xrightarrow{g} Q'$ for some $g \in E\backslash\{f\}$,
2. $C_{e,f}(Q) \xrightarrow{y} C''(Q')$ with $Q \xrightarrow{e} Q'$.

In 1, $S_{e,f} \xrightarrow{x}_\diamond S'_{e,f}$ obviously provides a match as $(C'_{e,f}(Q'), S'_{e,f}) \in \mathcal{R}$. In 2, we claim that $S_{e,f} \xrightarrow{y}_\diamond S''_{e,f}$ will provide a match as $C''_{e,f}(Q') \lhd S''_{e,f}$. To see this we only need to argue that $Q' \xrightarrow{f}$. Now let:

$$P \xrightarrow{e_0} \xrightarrow{e_1} \cdots \xrightarrow{e_{2n-1}} \xrightarrow{e_{2n}} Q \xrightarrow{e} Q'$$

be the odd length derivation sequence leading to Q' from some P which provides a strategy with respect to G. We have $Q' \xrightarrow{g}$ for any $g \notin \text{Follow}(e)\backslash\{e_0, \ldots, e_{2n}, e\}$ which includes f as $f \in \text{Follow}(e)\backslash\{e\}$ and all the edges $e_0, \ldots, e_{2n}$ are assumed to be different from f. For pairs $(C'_{e,f}(Q), S'_{e,f}) \in \mathcal{R}$ a similar arugument can be given.

$\Rightarrow$ Assume P does not provide a strategy with respect to G. Then there exists some odd length sequence:

$$P \xrightarrow{e_1} P_1 \xrightarrow{e_2} P_2 \cdots \xrightarrow{e_j} P_j$$

(i.e. j is odd) such that for some $e \in \text{Follow}(e_j)\backslash\{e_1, \ldots, e_j\}$, $P_j \not\xrightarrow{e}$. In this case we must have $C_{e_j,e}(P) \not\lhd S_{e_j,e}$,since the only possible match for:

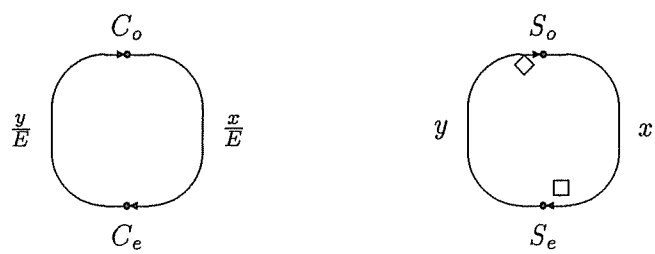
$$C_{e_j,e}(P) \xrightarrow{x} \cdots \xrightarrow{x} C_{e_j,e}(P_{j-1}) \xrightarrow{y} C''_{e_j,e}(P_j)$$

is

$$S_{e_j,e} \xrightarrow{x}_\diamond \cdots \xrightarrow{x}_\diamond S_{e_j,e} \xrightarrow{y}_\diamond S''_{e_j,e}$$

However, $C''_{e_j,e}(P_j) \not\xrightarrow{x}$ as $P_j \not\xrightarrow{e}$. Hence the requirement $S''_{e_j,e} \xrightarrow{x}_\square$ can not be met. □

Lemma A.5 *Let C_o and S_o have the following behaviours:*



where x and y are different actions. Then P is a solution to the (singleton) inequationsystem:

$$\mathcal{E}_o = \{(C_o, S_o)\}$$

if and only if all finite *computations of P have odd length.*

Proof:
$\Leftarrow$ First define the sets:

$$\begin{aligned} A_o &= \{P \mid \forall c \in \mathrm{Comp}(P).|c| \text{ is odd}\} \\ A_e &= \{P \mid \forall c \in \mathrm{Comp}(P).|c| \text{ is even}\} \end{aligned}$$

Then we show that the relation:

$$\mathcal{R} = \{(C_o(P), S_o) \mid P \in A_o\} \cup \{(C_e(P), S_e) \mid P \in A_e\}$$

is a refinement. First consider $(C_o(P), S_o) \in \mathcal{R}$ and assume $C_o(P) \xrightarrow{x} C_e(P')$ with $P \xrightarrow{e} P'$ for some e (this is the only possible type of transition for $C_o(P)$). As $P \in A_o$ obviously $P' \in A_e$ and $S_o \xrightarrow{x}_\Diamond S_e$ provides a matching move since $(C_e(P'), S') \in \mathcal{R}$. Now consider the required transition $S_o \xrightarrow{x}_\Box S_e$. As $P \in A_o$, $P \xrightarrow{e} P'$ for some e and $P' \in A_e$. But then $C_o(P) \xrightarrow{x} C_e(P')$ is a matching move. Secondly, consider $(C_e(P), S_e) \in \mathcal{R}$ and assume $C_e(P) \xrightarrow{y} C_o(P')$ with $P \xrightarrow{e} P'$ for some e. Obviously $P' \in A_o$, and hence $S_e \xrightarrow{y}_\Diamond S_o$ provides the matching move as $(C_o(P'), S_o) \in \mathcal{R}$. As S_e has no required transitions this concludes the verification of $\mathcal{R}$ being a refinement.

$\Rightarrow$ Assume $P \notin A_o$. That is P has some finite computation of even length:

$$P \xrightarrow{e_1} P_1 \xrightarrow{e_2} \cdots \xrightarrow{e_{2k}} P_{2k}$$

(i.e. $P_{2k} \not\xrightarrow{e}$ for all $e \in E$). But then:

$$C_o(P) \xrightarrow{x} C_e(P_1) \xrightarrow{y} \cdots \xrightarrow{y} C_o(P_{2k})$$

is a computation for $C_o(P)$. Now assume $C_o(P) \lhd S_o$. Then obviously we must have $C_o(P_{2k}) \lhd S_o$. However, as $C_o(P_{2k}) \not\xrightarrow{x}$, but $S_o \xrightarrow{x}_\Box$ we have a contradiction. □

We can now state and prove Theorem 3.1.

Theorem A.6 *The decision problems* InEq *and* $\textsc{InEq}_d$ *are both PSPACE–hard.*

Proof: It follows easily from Lemma A.2 – A.5 that P is a solution to the inequation system $\mathcal{E} = \mathcal{E}_R \cup \mathcal{E}_G \cup \mathcal{E}_S \cup \mathcal{E}_o$ if and only if P provides a winning strategy with respect to G. Thus PSPACE–hardness of InEq follows directly from PSPACE–completeness of GenGeo and the fact that the inequation system constructed by Lemma A.2 – A.5 has polynomial size relative to the original graph. PSPACE–hardness of $\textsc{InEq}_d$ follows by noting that the modal specifications of inequations constructed in Lemmas A.2 – A.5 are all deterministic.

□